HW: Safe Cons

- n procs
  - In a solo run return id
  - In not-solo can all return an id or an arbitrary uninterruptable value
- Election out of safe cons?
  - Trivial for 2 procs
    - If ``value'' output p0
  - What about 3 processors? Cannot settle on default because a non-participating processors is not allowed to be chosen

Solution to safe:

Induction on n.

Proc 1...n-1 decide PA (an id), using Hypo.

Procs 2...n decide PB using hypo.

Afterwards All invoke Safe cons using their ids.

If return is id-n return PB else PA.

Proof left to the reader. Google DBLP Eli Gafni ``tight group Renaming…’’

Dijkstra MX cannot be done ``wait''

- Cannot be done one-shot even for n=2.
  - It amounts to one outputs ``win'' and the other ``lose.''
  - HW: Use Konig Lemma to show any protocol the # rounds to output win bounded – others with no `win' by then output `lose'

- But MX is in the context of asynchronous Shared-Memory?

- Asynchrony = trying in synchronous round and not succeeding is asynchrony

Solution:

Konig Lemma says that an infinite nodes rooted tree with finite branching has an infinite depth.

A wait-free execution every step is an edge in a rooted tree. The tree has finite branching since the number of processors is finite. If for any integer b there is an execution longer than b then the number of nodes is infinite. By Konig there is an infinite path. But infinite path is an execution in which a processor does not decide.

DBLP Eli Gafni ``generalized imppossibility result…''

But MX is in SM and we are MP???

- In each round of Adv at least one message between every pair
- WHAT IS SHARED-MEMORY???
  - Obviously each read the other or both, but is that all?
    - E.g. in SM the first to write is read by all – MP not necessarily

- MP contains tournament, SM contains TRANSITIVE tournament (why?)
- If contains transitive tournament does there exist a schedule justifying?
- Can MP ``implement'' SM

- But MX is not round by round (HW: implement)

Solution:

Can implement Persistent SM round by round:

At round i, post your set of ``writes'', snapshot the other cells of round i. Each cell has a set of ``writes''. If the cardinality of the union of the ``writes'' is i return the set as snapshot add a new write to your set which is the result of what you just snaped, else (by necessity it will be more than i), go to i+1.

Prove the alg. Think what to do with the ``wait''.

DBLP Eli Petr opodis 2010 and Herlihy PODC 2010. (appears in two papers but only one ``inventor'' :))

Leader Election Version of Cons

- p0,p1, two independent cons, pi solo, output pi, else output same in every cons.
- This generalizes to 3
  - Can 3 procs 3 independent cons agree in at least 1, if can, agree in at least 1 out of 2?

- HW: Show (n,j) leader election = n procs, 1 out of j cons.
  - (n,j) leader election: each outputs participating member (heard about) and |{outputs}|< j+1

(n,j) => 1 out of j cons:

Post a value for cons1, read, if only same value posted return it. Else choose the highest value and go to cons2.

Lemma: the lowest value from (n,j) will not proceed to cons2.

<=

Just output what you got from 1 out of j cons.

HW show still holds if 1 out of j cons is binary cons.

(2 independent cons return from 1) = (3 processors output cumulatively <3 ids) (HW!).

- Each submits its id as proposed decision vector value to cons 1 and cons 2.
- Return an id from any of the 2 cons that returned.
- Since cons returns a value submitted, induces Sperner coloring on the subdivided simplex
- There must be a fully colored triangle
- 3 processors with the possibility of a message loss cannot solve 2-set cons!

Looks like a corollary of previous.

If Adv can control x connections, how high value of set-cons it can force?

- If can control all n choose 2 connections = r/w (sperner)
  - What if (n choose 2)-1?
  - In each round some 2 procs exchange messages
  - Do snapshot* + round
  - In ``round'' proc with small snap adopts larger snap
  - At least 2 procs return the same snap
  - Each proc returns max id in its snap
  - |returns|<n   n-1 set cons!

- HW: Extend to all values of x

Solution:

You can extend the alg according to how many Strongly connected components (by edges the adversary will not touch) the adversary can create.

Go in the reverse. The Adversary contronl nothing. Single component. To create 2 components it needs n-1 edges so can disconnect one node from all. To create 3 components control 2(n-2) to isolate 2 from the rest then add another control to isolate the 2 from each other. Continue inductively.

Essentially (found belatedly :)) in DBLP Emmanuel Godard OPODIS 2016

Extend (n,j) cons to j state-machines at least one advance

- Simple use of CA (HW?)
- The State-Machines can be read-write threads reading and writing to each other
- n procs with j-set cons can advance at least one out of j threads.
- (n,j)-set cons = j concurrency

This should have come after commit adopt..

See DBLP Eli Gafni ``Generalized Universality'' CAV 2011